Validation Tool Code Walk-through

Engineering Node

June 21-22, 2006

http://pds.nasa.gov

Agenda - Day 1

8:30 - 8:35 AM	**Opening Remarks**	Dan Crichton
8:35 - 9:05 AM	**PDS Architecture**	Dan Crichton
9:05 - 9:50 AM	**Standards Overview**	Elizabeth Rye
9:50 - 11:30 AM	**Validation Tool Requirements and Design**	Sean Hardman Paul Ramirez
11:30 - 12:00 PM	**Code Walk-through Guidelines**	Sean Kelly
1:00 - 5:00 PM	**Validation Tool Code Walk-through**	Paul Ramirez

Agenda - Day 2

8:30 - 8:35 AM	**Opening Remarks**	Dan Crichton
8:35 - 9:45 AM	**Test Approach**	Emily Law
9:45 - 10:30 AM	**Questions**	All
10:30 - 12:00 AM	**Wrap Up / Action Items**	Dan Crichton

Opening Remarks

Dan Crichton

Welcome

- Introductions
- Logistics
- Goals
- Questions

PDS Architecture

Dan Crichton

Outline

- PDS Mission
- Software Architecture Definition and PDS
- Critical PDS Architectural Tenets
- Level 1 Requirements
- OAIS Functional Archive Model
- Basic Ingest Concepts
- Critical Level 2/3 Requirements
- PDS Data Architecture and Standards
  - High Level Data Model
  - High Level Product Model
  - Data Standards
- Existing PDS Validation Tool
- Guiding Documents

PDS Mission

The mission of the Planetary Data System is to facilitate achievement of NASA's planetary science goals by efficiently collecting, archiving, and making accessible digital data produced by or relevant to NASA's planetary missions, research programs, and data analysis programs*

*July 2005 e-vote by PDS Management Council

Software System Architecture and PDS

- Architecture: The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution. (ANSI/IEEE Std. 1471-2000)

- What does the PDS Architecture contain?
  - Architectural Descriptions
  - Architectural Principles/Tenets
  - Models (data and software architecture)
  - Standards

- PDS is a geographically distributed data system with nodes consisting of 7 Discipline Nodes (Atmospheres, Geosciences, Imaging, Navigation, PPI, Rings, Small Bodies), 1 Function (RS), 1 Program Management Node, 1 Engineering Node

Critical Architectural Tenets Applied by EN

- Model-driven: The data model and associated data standards, controlled via the PDS Standards process, drives the system
- Consistent Data Elements: Data elements are applied and used consistently across the system
- Independence: The system is independent of any mission, node, platform or location
- Reuse: Both software and data are defined and implemented in such a way as to be reused in different deployments
- Open Source: Open source components are adopted, where available
- Resource Impact: Adoption of PDS tools, components and standards have minimal resource impacts on nodes and missions

PDS Level 1 Requirements

1. PDS will provide expertise to guide and assist missions, programs, and individuals to organize and document digital data supporting NASA's goals in planetary science and solar system exploration.

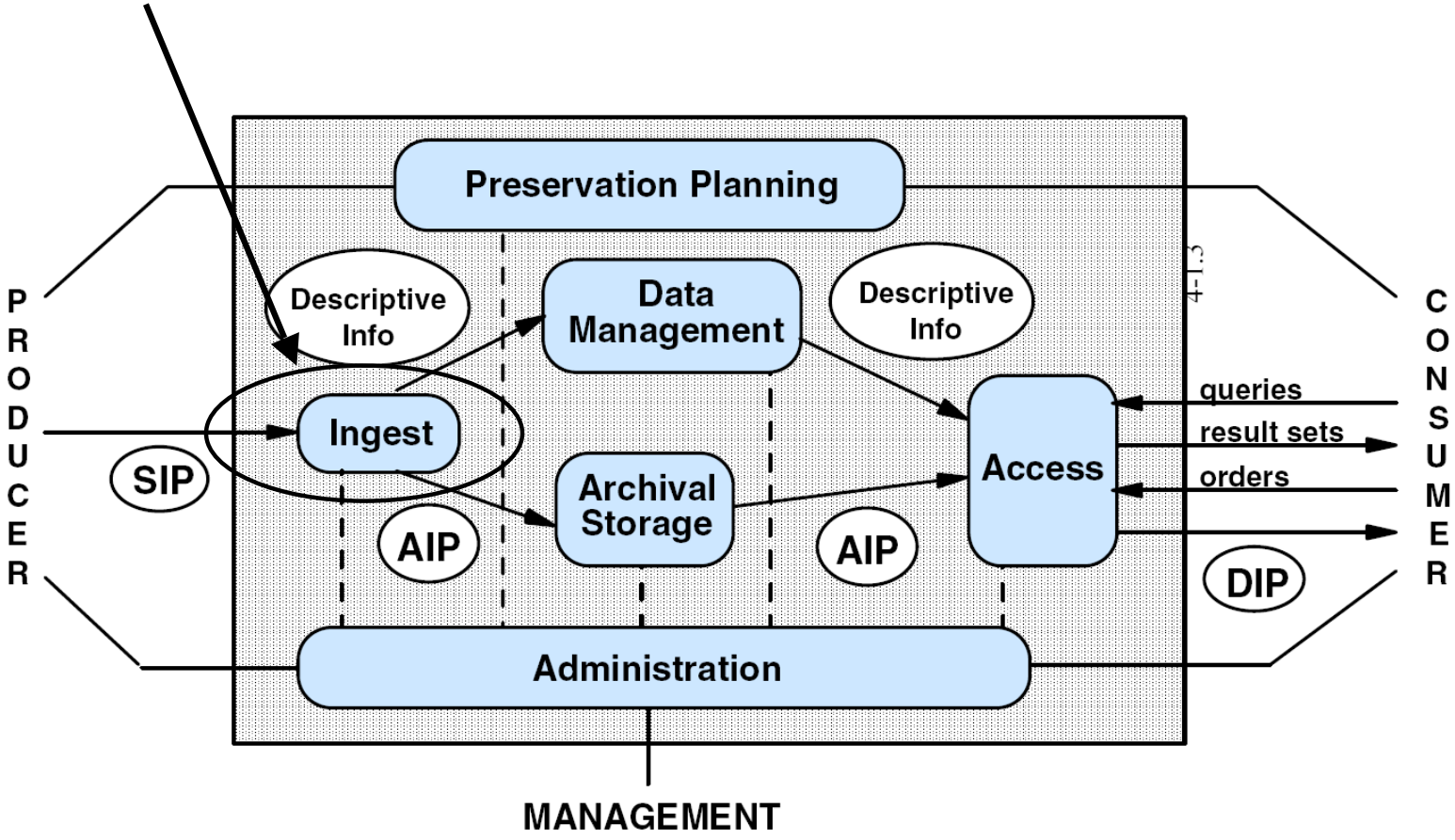
2. PDS will collect suitably organized and well-documented data into archives that are peer reviewed and maintained by members of the scientific community.

3. PDS will make these data accessible to users seeking to achieve NASA's goals for exploration and science.

4. PDS will ensure the long-term preservation of the data and maintain their usability.

Archiving Functional Model

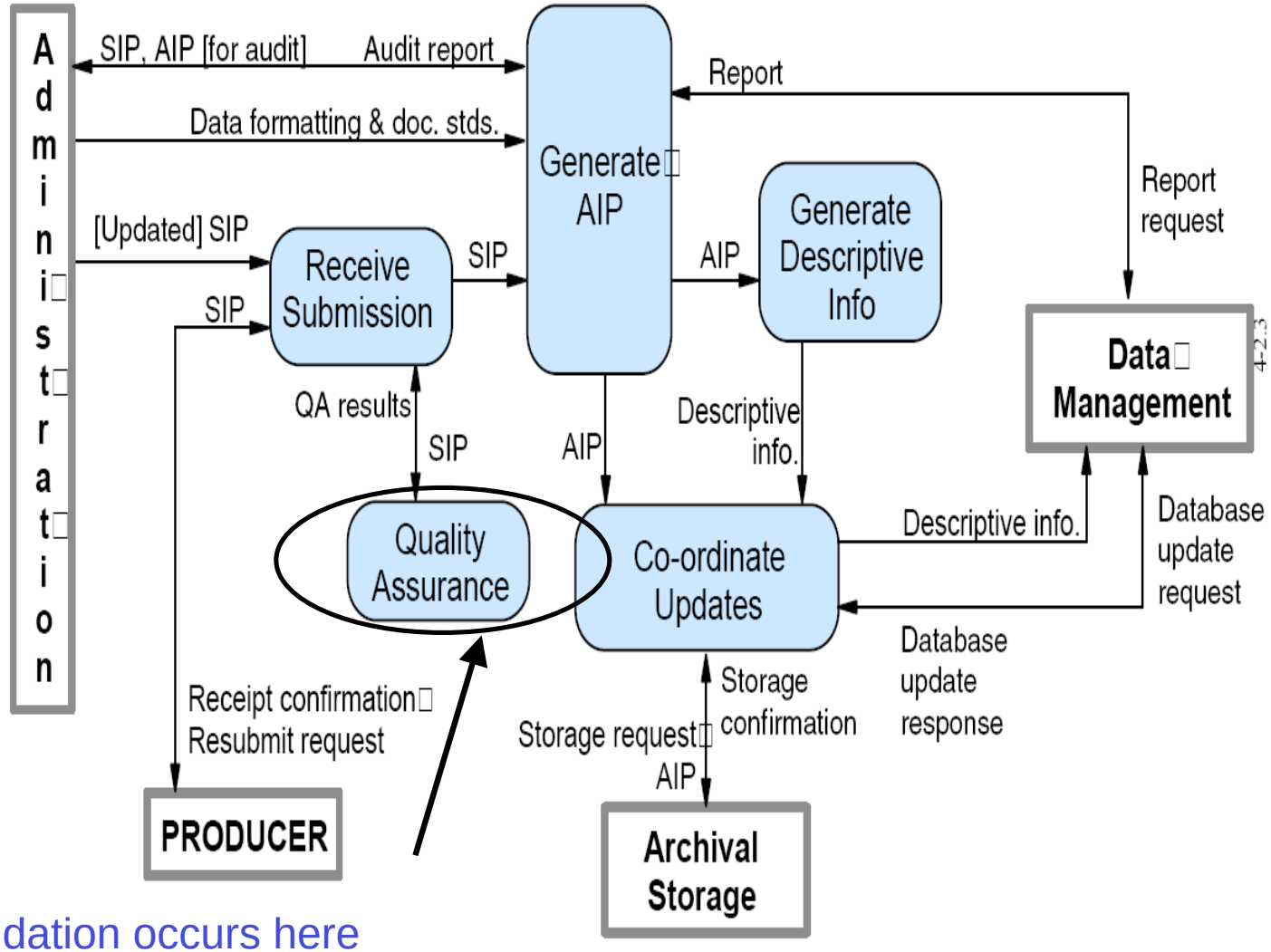
Validation occurs here



NOTE: OAIS was developed from PDS. There are some differences, but can be used to explain concepts.

Reference Model for Open Archive Information System, CCSDS 650.0-B-1, January 2002

OAIS Archiving Ingest Model



Validation occurs here

Reference Model for Open Archive Information System, CCSDS 650.0-B-1, January 2002

Basic PDS Ingest Concepts

- PDS Data *Producers* include missions, instrument teams, PIs, etc who submit data to PDS for inclusion in the archive

- PDS Data *Producers* require
  - Standards for archival products
  - Tools for validating archival products
  - Mechanisms for submitting data products

- PDS Discipline Nodes coordinate data submissions
  - Work with producers to design data products based on PDS Standards
  - Schedule the delivery of products
  - Validate all products submitted to the archive

Level 2/3 Archiving Standards

1.4 Archiving Standards: PDS will have archiving standards for planetary science data

1.4.1 PDS will define a standard for organizing, formatting, and documenting planetary science data

1.4.2 PDS will maintain a dictionary of terms, values, and relationships for standardized description of planetary science data

1.4.3 PDS will define a standard grammar for describing planetary science data

1.4.4 PDS will establish minimum content requirements for a data set (primary and ancillary data)

1.4.5 PDS will establish minimum sets of archival data from missions and other major data providers

1.4.6 PDS will develop, publish and implement a process for managing changes to the archive standards

1.4.7 PDS will keep abreast of new developments in archiving standards

Level 2/3 Tool Requirements

1.5 Archiving Tools: PDS will have tools to assist data producers in assembling, validating, and submitting archival products

1.5.1 PDS will provide tools to assist data producers in generating PDS compliant products

1.5.2 PDS will provide tools to assist data producers in validating products against PDS standards

1.5.3 PDS will provide tools to assist data producers in submitting products to the PDS archive

1.5.4 PDS will provide documentation for installing, using, and interfacing with each tool

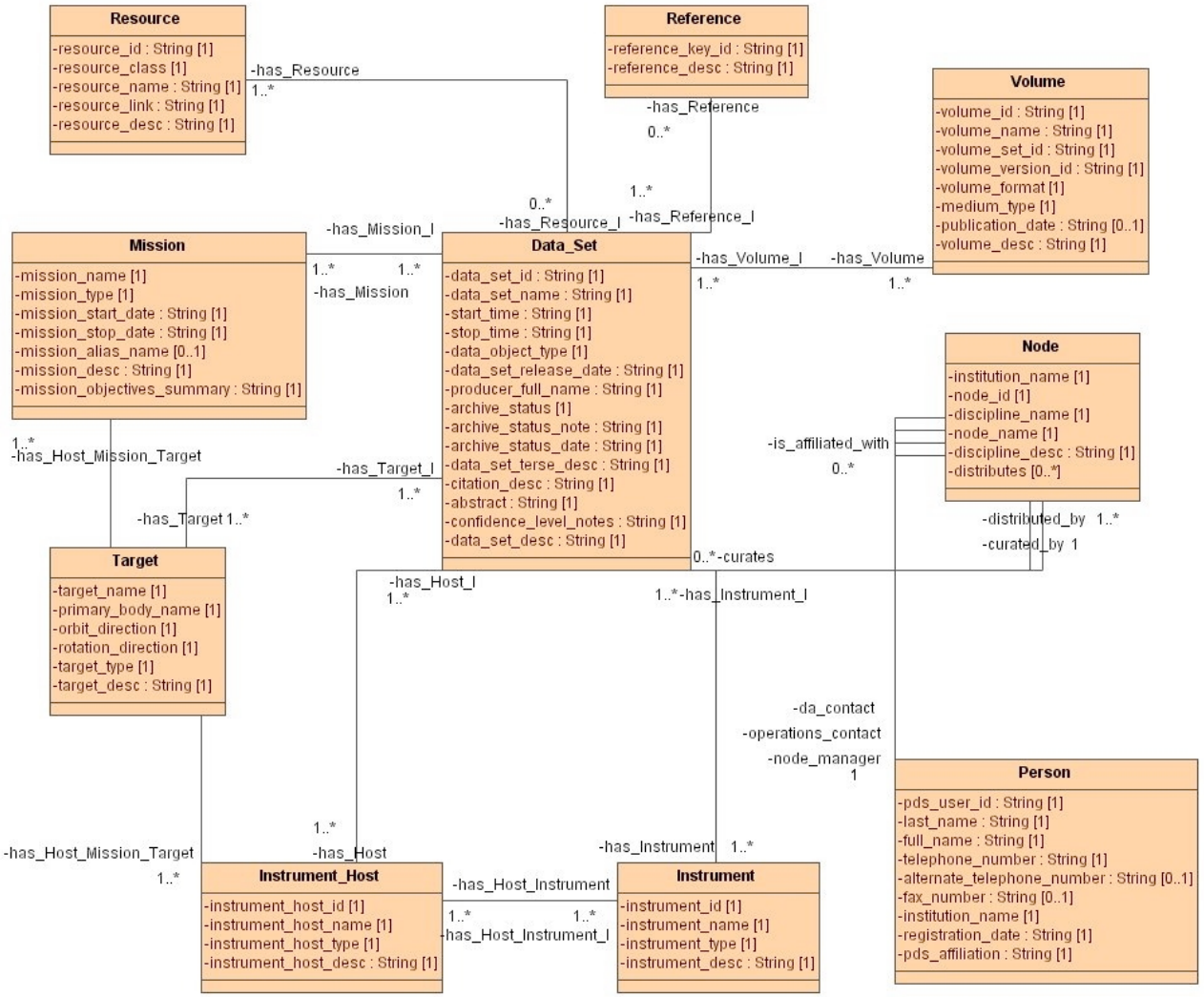
Level 2/3 Validation Requirements on Data Submissions to PDS

2.2 Validation: PDS will validate data submissions to ensure compliance with standards.

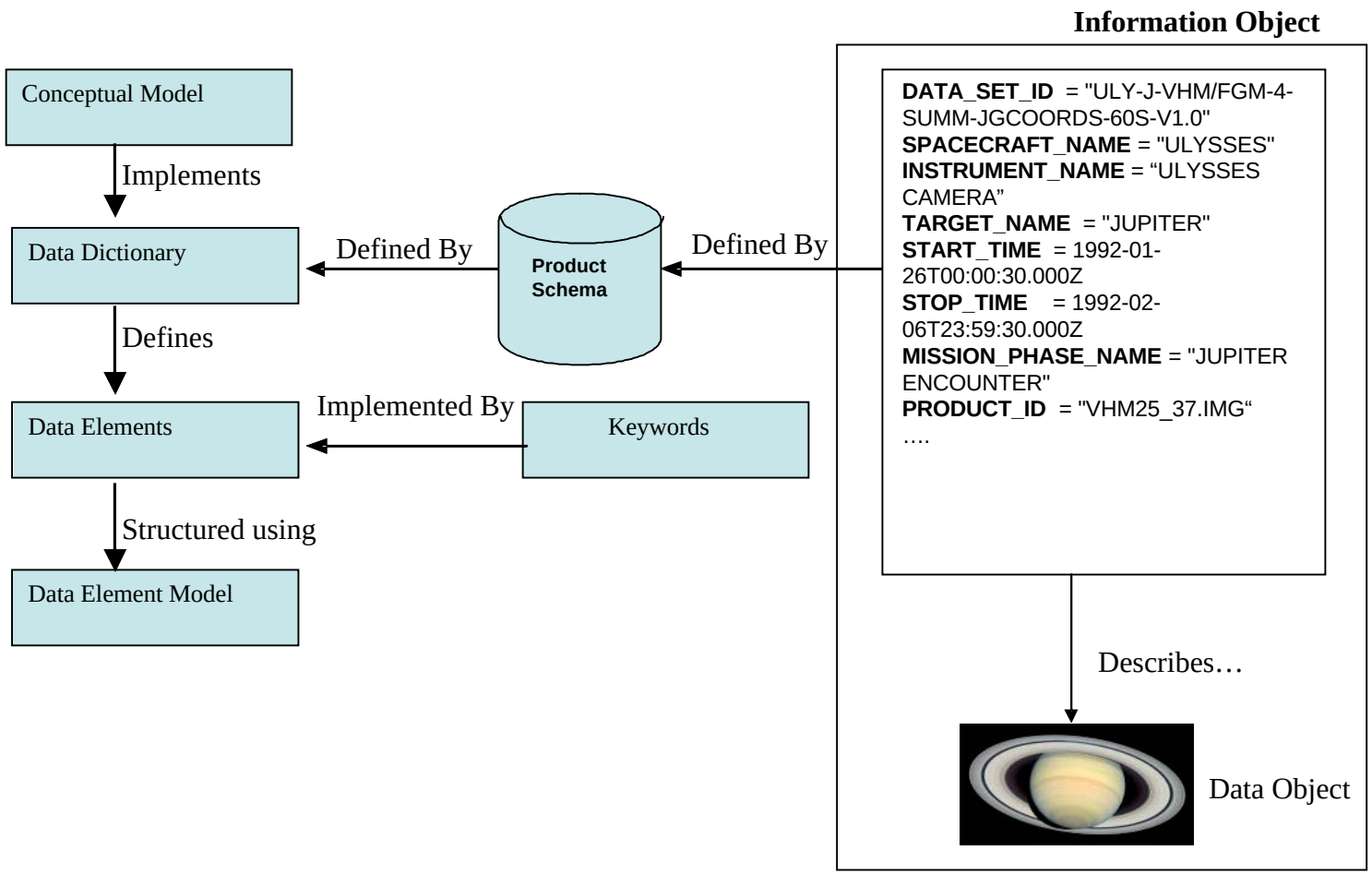
2.2.1 PDS will develop and publish procedures for determining syntactic and semantic compliance with its standards

2.2.2 PDS will implement procedures to validate all data submissions to ensure compliance with standards

PDS High Level Conceptual Data Model



PDS Conceptual Product Model Architecture



Metadata Object (in ODL)

Data Standards

- Planetary Science Data Dictionary

- Standards for organizing archival data
  - Volume standards (e.g., Layout, catalog files, etc)
  - Structure/grammar standards (e.g, Object Description Language)
  - Standards for describing data products (e.g., data object structure, etc)

Existing PDS Validation Tool

- LVTOOL: Developed in the early 1990s to validate PDS archives
  - Used extensively by the nodes
  - Used sometimes by the missions
  - Ported to various platforms
  - Increasing number of issues
  - Original developers long gone…
  - Needs and experience of PDS has evolved

Guiding Documents

- PDS Level 1,2,3 Requirements, May 2006
- Reference Model for Open Archive Information System, CCSDS 650.0-B-1, January 2002
- Reference Architecture for Space Information Management (draft), CCSDS 312.0-G-1, May 2006
- Planetary Data System (PDS) Standards Reference, March 20, 2006, Version 3.7, JPL D-7669, Part 2
- Planetary Science Data Dictionary Document, August 28, 2002, Planetary Data System (PDS), JPL D-7116, Rev E.
- Information technology—Metadata registries (MDR)—Part 1: Framework. International Standard, ISO/IEC 11179-1:2004. 2nd ed. Geneva: ISO, 2004

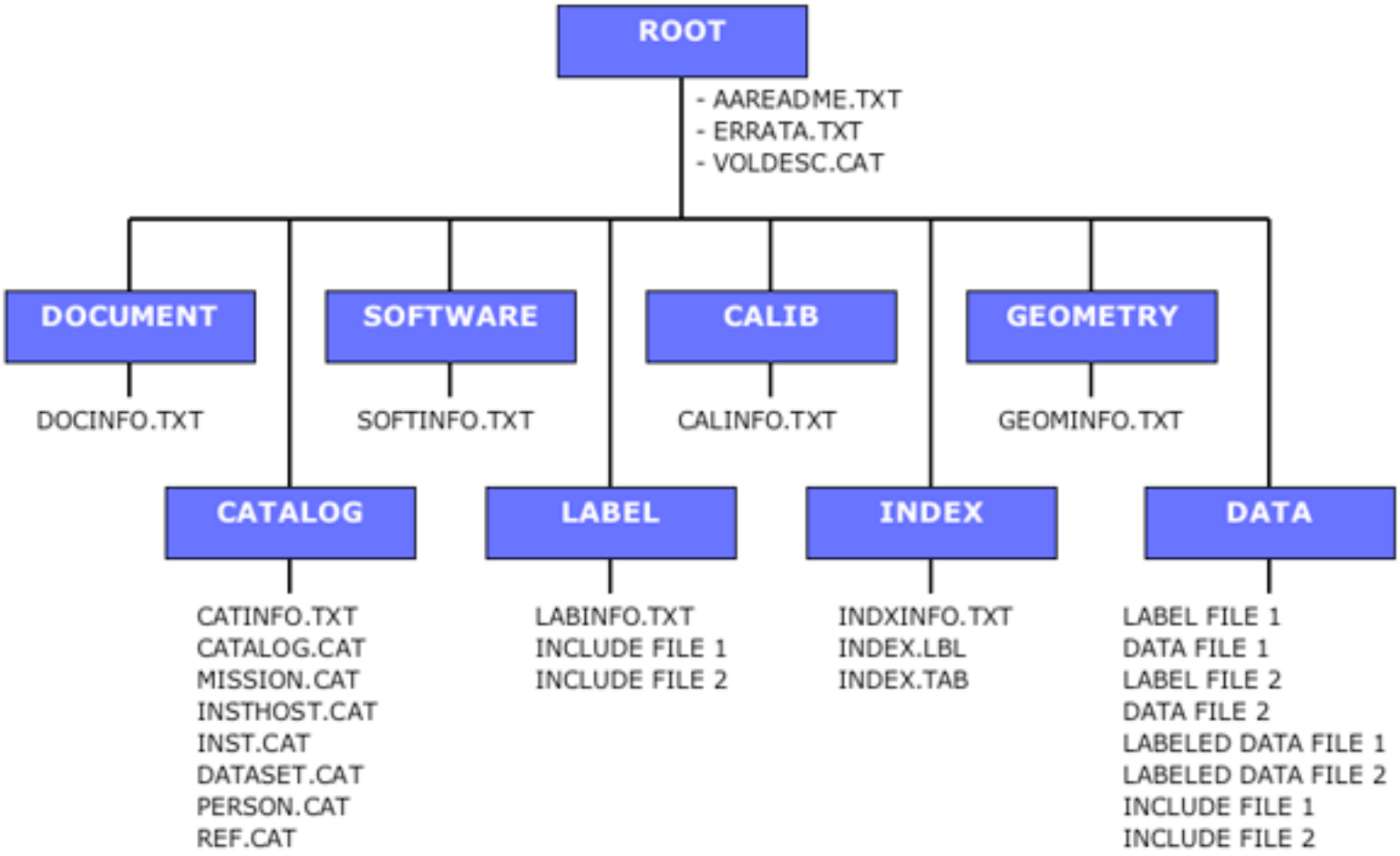
Standards Overview

Elizabeth Rye

Outline

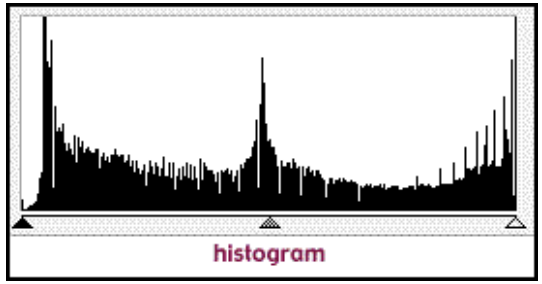
- Brief overview of an archive volume / package
- What is a data product?
- What is a data object?
- What is a PDS label?
  - Grammar - the Object Description Language
  - Nomenclature - Data Element and Object Names and the PSDD
  - Format - Relationships between files, labels, and data objects
  - Contents - Expected components of a PDS label

PDS Volume Structure



What is a PDS Data Product?

```
PDS_VERSION_ID = PDS3
DD_VERSION_ID =
LABEL_REVISION_NOTE =

/* FILE CHARACTERISTICS */
RECORD_TYPE = FIXED_LENGTH
RECORD_BYTES = 324
FILE_RECORDS = 334
LABEL_RECORDS = 7

/* POINTERS TO DATA OBJECTS */
^IMAGE = 8
^HISTOGRAM = 333

/* IDENTIFICATION DATA ELEMENTS */
  .
  .
  .

/* DATA OBJECT DEFINITIONS */
OBJECT = IMAGE
  .
  .
  .
END_OBJECT = IMAGE

OBJECT = HISTOGRAM
  .
  .
  .
END_OBJECT = HISTOGRAM
END
```

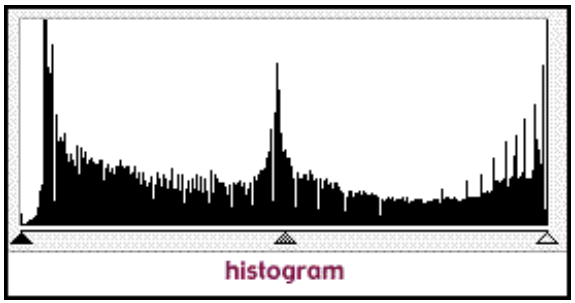


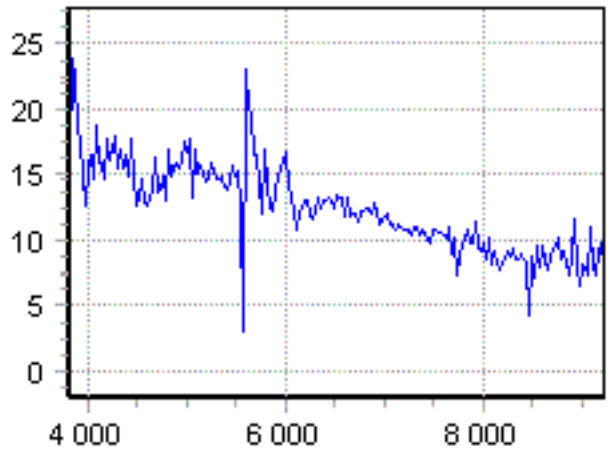
What is a Data Object?

- Primary Objects
  - IMAGE
  - QUBE
  - SERIES
  - SPECTRUM
  - SPREADSHEET
  - TABLE

	301	46	184.0	4.0	36.0
6	18 25 36.343	227.42	227.42	227.06	224.62
6	18 25 40.236	227.42	227.51	227.18	224.61
6	18 25 44.130	227.71	227.97	227.43	224.80
6	18 25 48.023	227.16	227.37	227.16	224.71
6	18 25 51.917	227.12	227.31	227.24	225.01
6	18 25 55.810	227.46	227.56	227.56	225.21
6	18 25 59.703	227.50	227.66	227.59	225.21
6	18 26 3.597	227.65	227.77	227.59	225.30
6	18 26 7.490	227.78	227.72	227.44	225.50
6	18 26 11.383	227.47	227.60	227.53	225.28

- Secondary Objects
  - HEADER
  - HISTOGRAM
  - PALETTE





What is a PDS Label?

- Grammar
- Nomenclature
- Format
- Contents

Object Description Language

The Object Description Language (ODL) is the language used to encode data labels for the Planetary Data System.

- Character Set
- Lexical Elements
- Statements
- Values
- ODL / PVL Usage

Object Description Language

- Character Set
  - ISO 646; US version is ASCII
  - Letters - A-Z, a-z; not case sensitive (except for character strings)
  - Digits - 0-9
  - Special Characters - = {} () + - <> . “” ‘’ _ , / * : # & ^
  - Spacing Characters - space (horizontal tab)
  - Format Effectors - CR, LF (FF, VT)
  - Control characters - may appear in text strings

Object Description Language

- Lexical Elements
  - Lexical elements are the basic building blocks of the ODL. Statements in the language are composed by stringing lexical elements together. The lexical elements of ODL are:
    - Numbers
      - Integers in decimal notation (123)
      - Integers in based notation (2#1111011#)
      - Real numbers in decimal notation (123.4)
      - Real numbers in scientific notation (1.234E2)
    - Dates and Times
      - Dates (YYYY-MM-DD or YYYY-DOY)
      - Times (hh:mm:ss[.sss][Z]) (All times interpreted as UTC)
      - Date/Times (YYYY-MM-DDThh:mm:ss[.sss]Z)
      - Date/Time formats may be truncated to match precision of the value

Object Description Language

- Lexical Elements (cont'd)
  - Strings
    - Text strings
      - » Double quoted
      - » Used to hold arbitrary strings of characters, including format effectors and control characters; thus may span multiple lines
      - » May not contain double quote
    - Symbol strings
      - » Single quoted
      - » Sequences of characters used to represent symbolic values (ex. Image_id = 'J123-U2A')
      - » May not contain apostrophe, format effectors, or control characters; thus, may *not* span multiple lines
    - Current PDS implementation of ODL does not require syntactic differentiation between symbols and text strings; therefore, double quotes should be used in both cases

Object Description Language

- Lexical Elements (cont'd)
  - Identifiers
    - Used as the names of objects and attributes, and as the value of symbolic literals
    - Composed of letters, digits, and underscores
    - Underscores are used to separate words
    - First character must be a letter; the last character may not be an underscore
    - Examples: VOYAGER_2, BLUE_FILTER, USA_NASA_PDS_1_0007
    - Reserved Identifiers: END, END_OBJECT, END_GROUP, GROUP, OBJECT
  - Special symbols used for operators, etc.
    - = the assignment operator
    - , separates the values of an array or a set
    - * serves as the multiplication operator in units expressions
    - / serves as the division operator in units expressions
    - ^ denotes a pointer to an object
    - < > enclose units expressions
    - ( ) enclose the elements of a sequence
    - { } enclose the elements of a set
    - ** the exponentiation sign within units expressions

Object Description Language

- Statements
  - An ODL-encoded label is made up of zero, one, or more statements followed by the reserved identifier END.
  - There are four types of statements:
    - Attribute assignment statement
    - Pointer statement
    - Object statement
    - Group statement

Object Description Language

- Statements (cont'd)
  - Labels are composed of lines; each line is a string of characters terminated by the CR, LF format effectors
    - Only one statement per line, although a statement may span multiple lines
    - Format effectors may appear before, after, or between lexical elements without changing the meaning of the statement. Thus, the following are equivalent:

```
FILTER_NAME = {RED, GREEN, BLUE}
FILTER_NAME = {RED,
               GREEN,
               BLUE}
```

    - A line may include a comment
      - Comments may not span lines
      - Comment delimiters are "/*" and "*/"
      - Comments may not contain format effectors
      - Comments should be ignored when parsing ODL labels
      - Comment delimiters within text strings are interpreted as part of string
      - Characters on a line following a comment are ignored

Object Description Language

- Statements (cont'd)
  - Attribute Assignment Statements
    - Used to specify the value for an attribute of an object
    - Value may be a singular scalar value, an ordered sequence of values, or an unordered set of values
    - May optionally contain a namespace identifier; when pre-pended to the element identifier, indicates that the latter has a local definition within the context identified by the namespace_identifier
    - Examples:

```
RECORD_BYTES         = 800
TARGET_NAME          = JUPITER
SOLAR_LATITUDE       = (0.25 <DEG>, 3.00 <DEG>)
FILTER_NAME          = {RED, GREEN, BLUE}
CASSINI:TARGET_NAME  = JUPITER
MRO:SOLAR_LATITUDE   = (0.25 <DEG>, 3.00 <DEG>)
VOYAGER:FILTER_NAME  = {RED, GREEN, BLUE}
```

Object Description Language

- Statements (cont'd)
  - Pointer Statements
    - Indicate the location of an object
    - Value may be a scalar, ordered sequence, or unordered set
    - Within PDS, three types of pointers:
      - Data Location Pointers
        - » Indicate the position of an object within another object
        - » May refer to another location within the same file or to a location within an external file
        - » If units not specified, default is records

```
^IMAGE          = 12
^IMAGE          = 600 <BYTES>
^INDEX_TABLE    = "INDEX.TAB"
^SERIES         = ("C100306.DAT", 2)
^SERIES         = ("C100306.DAT", 700 <BYTES>)
```

- Statements (cont'd)
  - Pointer Statements (cont'd)
    - Another common usage of pointers is to reference external files
      - Include Pointers
        - » Files referenced by include pointers are included directly at the location of the pointer statement
        - » Act like the "#include" statements in C program source files

```
^STRUCTURE                = "ENGTAB.FMT"
^STRUCTURE                = "IMAGE.FMT"
^CATALOG                  = "CATALOG.CAT"
^DATA_SET_MAP_PROJECTION  = "DSMAPDIM.CAT"
```

      - Related Information Pointers
        - » Files references by related information pointers provide additional documentation of special use to human readers
        - » Formed using elements ending in "DESCRIPTION" or "DESC"
        - » Reference text files not written in ODL

```
^DESCRIPTION              = "TRK_2_25.ASC"
```

Object Description Language

- Statements (cont'd)
  - Object Statements
    - The OBJECT statement begins the description of an object
    - The object description consists of a set of attribute assignment statements defining the values of the object's attributes
    - If the object is itself composed of other objects, then OBJECT statements for the component objects are nested within the object's description

```
OBJECT       = object_identifier
  ATTRIBUTE1 = VALUE1
  ATTRIBUTE2 = VALUE2
END_OBJECT   = object_identifier
```

    - The object identifer gives a name to the particular object being described. (Ex. IMAGE and BROWSE_IMAGE)
    - The object identifier at the end of the OBJECT statement is optional, but if present, must match that at the beginning.

Object Description Language

- Statements (cont'd)
  - Group Statements
    - The GROUP statement is used to group together statements that are not components of a larger object (Ex. BAND_BIN group)

```
GROUP          = group_identifier
  ATTRIBUTE1   = VALUE1
  ATTRIBUTE2   = VALUE2
END_GROUP      = group_identifier
```

    - The group identifier gives a name to the particular group being described.
    - The group identifier at the end of the GROUP statement is optional, but if present, must match that at the beginning.
    - Although ODL permits nesting of groups, the PDS implementation does not. PDS groups may only contain attribute assignment statements, include pointers, or related information pointers (not data location pointers)

Object Description Language

- Values
  - ODL provides scalar values, ordered sequences of values, and unordered sets of values
  - A scalar value consists of a single lexical element, either numeric, date/time, text string, or symbolic
    - Numeric scalar values may optionally include units expression
      - A units expression is always included within angle brackets
      - The expression may consist of a single units identifier (<KM> for kilometers) or more complex values (<KM/SEC> for velocity)
      - There is no defined maximum or minimum magnitude or precision for numeric values since the actual range and precision of numbers that can be represented varies based on the computer platform being used. If software for reading ODL encounters a numeric value too large to be represented, the software must report an error to the user.

Object Description Language

- Values (cont'd)
  - Text string scalar values read in from ODL labels are reassembled into a string of characters, by replacing format effectors with space characters (exception when last character is a hyphen)
  - Symbolic values may be specified as either identifiers or symbol strings

```
TARGET_NAME       = IO
SPACECRAFT_NAME   = VOYAGER_2
SPACECRAFT_NAME   = "VOYAGER-2"
SPACECRAFT_NAME   = "VOYAGER 2"
REFERENCE_KEY_ID = SMITH1997
REFERENCE_KEY_ID = "LAUREL&HARDY1997"
```

  - Note that symbolic values are converted to uppercase on input.

"Voyager_2" ⟶ "VOYAGER_2"

Object Description Language

- Values (cont'd)
  - A sequence represents an ordered set of values. It can be used to represent arrays and other kinds of ordered data. Only one- and two-dimensional sequences are allowed.
  - A sequence may have any kind of scalar value for its members. It is not required that all members of the sequence be of the same type.

```
AVERAGE_ECCENTRICITY = (0, 1, 2, 3, 4, 5, 9)
INSTRUMENT_TEMPERATURE = ((34.7, 45.5), (23.8, 19.5))
```

  - Sets are used to specify unordered values drawn from some finite set of values. Note that the empty set is allowed.
  - The order in which the values appear in the set is not significant and need not be preserved when a set is read and manipulated. There is no upper limit on the number of values in a set.

```
FILTER_NAME = {RED, GREEN, BLUE, HAZEL}
```

What is a PDS Label?

- Grammar
- Nomenclature
- Format
- Contents

PDS Data Elements

The PDS has established data nomenclature standards which define the rules for constructing Data Element and Data Object names.

- Construction of Data Element Names
  - Composed of descriptor words and class words, connected by an underscore
  - Constructed from left to right, from most specific to most generic
    - "the name of a parameter in a data set"

      `DATA_SET_PARAMETER_NAME`
  - Constructed from standard ASCII alphanumeric characters and the underscore
  - Not case sensitive

PDS Data Elements

- Class Words
  - Comprise the rightmost component in a data element name
  - Identifies the basic information type
  - COUNT, DATE, DESCRIPTION, FLAG, FORMAT, GROUP, ID, MASK, NAME, NOTE, NUMBER, RANGE, RATIO, SEQUENCE, SET, SUMMARY, TEXT, TIME, TYPE, UNIT, VALUE, and VECTOR
- Descriptor Words
  - Should be selected from list of existing descriptors; new descriptor words can be submitted for review
  - Examples: ANGLE, ALTITUDE, LOCATION, RADIUS, and WAVELENGTH
- Range-Related Components
  - FIRST, LAST, START, STOP, MINIMUM, and MAXIMUM

PDS Data Elements

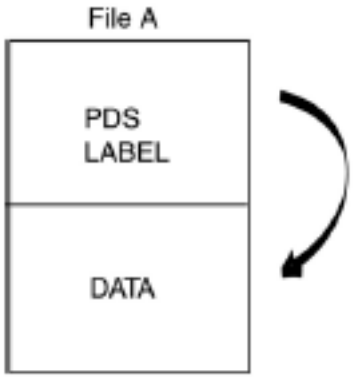
- Planetary Science Data Dictionary
  - Approved data elements and their attributes
    - General data type, unit id, standard value type, minimum and maximum value and length, default value, description, standard values, change date, status, source, etc.
  - Approved data objects
    - Description, required and optional data elements, required and optional sub-objects.

What is a PDS Label?

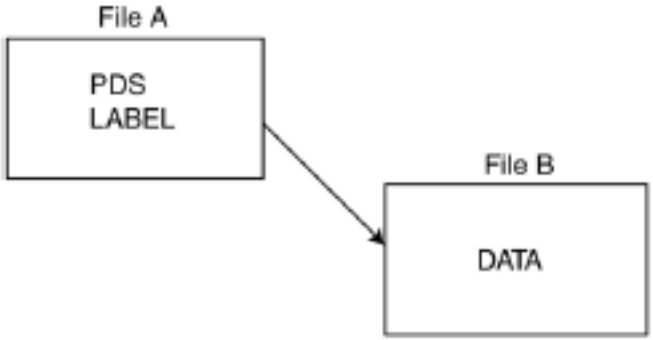
- Grammar
- Nomenclature
- Format
- Contents

Label Format

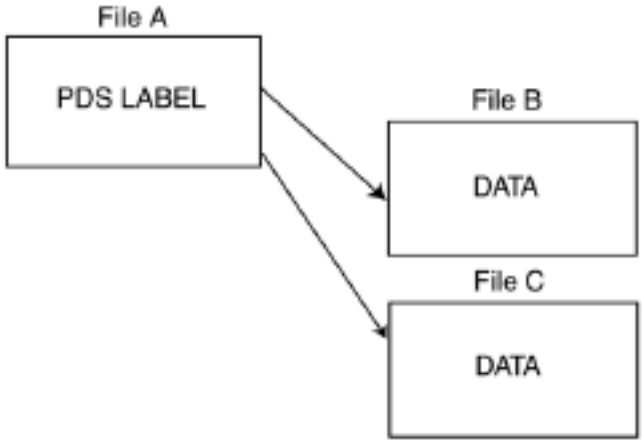
Attached



Detached



Combined
Detached



What is a PDS Label?

- Grammar
- Nomenclature
- Format
- Contents

Label Contents

- Label Standards Identifiers
- File Characteristics Data Elements
- Data Object Pointers
- Identification Data Elements
- Descriptive Data Elements
- Data Object Definitions
- END Statement

PDS LABEL

```
PDS_VERSION_ID                =
DD_VERSION_ID                 =
LABEL_REVISION_NOTE           =

/* FILE_CHARACTERISTICS */
RECORD_TYPE                   =
RECORD_BYTES                  =
FILE_RECORDS                  =
LABEL_RECORDS                 =

/* POINTERS TO DATA OBJECTS */
^IMAGE                        =
^HISTOGRAM                    =

/* IDENTIFICATION DATA ELEMENTS */
DATA_SET_ID                   =
PRODUCT_ID                    =
SPACECRAFT_NAME               =
INSTRUMENT_NAME               =
TARGET_NAME                   =
START_TIME                    =
STOP_TIME                     =
      .
      .
      .
PRODUCT_CREATION_TIME         =

/* DESCRIPTIVE DATA ELEMENTS */
FILTER_NAME                   =
OFFSET_MODE_ID                =
      .
      .
      .

/* DATA OBJECT DEFINITIONS */
OBJECT                        = IMAGE
      .
      .
      .
END_OBJECT                    = IMAGE
OBJECT                        = HISTOGRAM
      .
      .
      .
END_OBJECT                    = HISTOGRAM

END
```

Label Contents

- Label Standards Identifiers
  - Each PDS label must begin with the PDS_VERSION_ID data element. This identifies the version of the Standards to which the label adheres.
  - The PDS does not require Standard Formatted Data Unit (SFDU) labels on individual products, but they may be desired for conformance with specific project or other agency requirements. When provided, the SFDU label must *precede* the PDS_VERSION_ID keyword.
  - The DD_VERSION_ID element identifies the version of the PSDD to which a label complies.
  - The LABEL_REVISION_NOTE element is a free form, unlimited-length character string providing information regarding the revision status and authorship of a PDS label.

```
CCSD…                          [optional SFDU label]
PDS_VERSION_ID       = PDS3
DD_VERSION_ID        = PDSCAT1R52
LABEL_REVISION_NOTE = “1999-08-01, Anne Raugh (SBN), initial
   release.”
```

Label Contents

- File Characteristics Data Elements
  - The file characteristic data elements provide important attributes of the physical structure of a data product file. The data elements are:
    - RECORD_TYPE - the record characteristics of the data product file
    - RECORD_BYTES - the number of bytes in each physical record in the data product file
    - FILE_RECORDS - the number of physical records in the file
    - LABEL_RECORDS - the number of physical records that make up the label

Label Contents

- File Characteristics Data Elements (cont'd)
  - Not all of these data elements are required in every data product label. The table below lists the required (Req) and optional (Opt) file characteristic data elements for a variety of data products and labeling methods for both attached (Att) and detached (Det) labels. Where (max) is specified, the value indicates the maximum size of any physical record in the file.

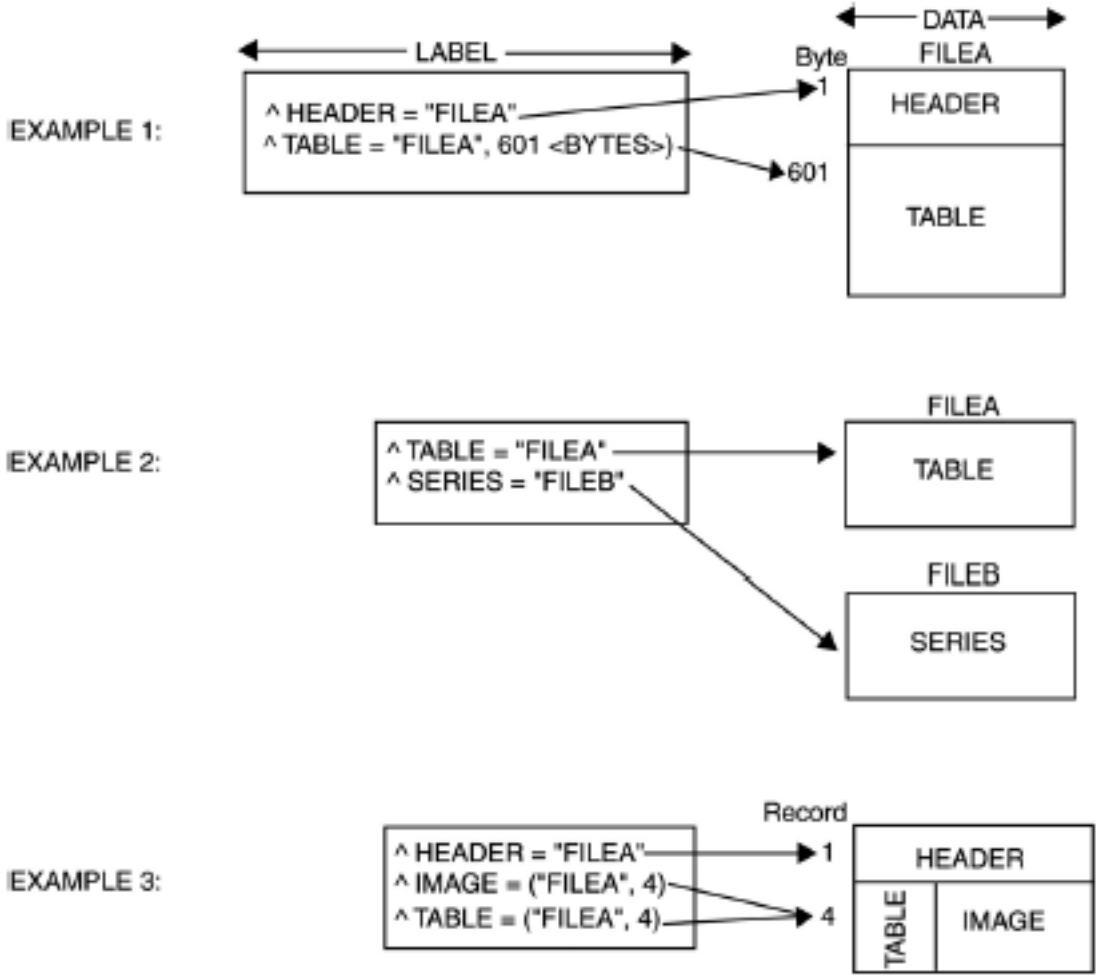
Labeling Method	Att	Det	Att	Det	Att	Det	Att	Det
RECORD_TYPE	FIXED_LENGTH		VARIABLE_LENGTH		STREAM		UNDEFINED	
RECORD_BYTES	Req	Req	Rmax	Rmax	Omax	n/a	n/a	n/a
FILE_RECORDS	Req	Req	Req	Req	Opt	Opt	n/a	n/a
LABEL_RECORDS	Req	n/a	Req	n/a	Opt	n/a	n/a	n/a

Label Contents

- Data Object Pointers
  - Data object pointers are required in labels with one exception: attached labels that refer to only a single object. In the absence of a pointer, the data object is assumed to start in the next physical record after the PDS product label area. This is commonly the case with ASCII text files described by a TEXT object and ASCII SPICE files described by a SPICE_KERNEL object.
  - Object pointers are required for all data objects, even when multiple data objects are stored in a single data product file.

Label Contents

- Data Object Pointers (cont'd)



Label Contents

- Identification Data Elements
  - These data elements provide additional information about a data product that can be used to relate the product to other data products from the same data set or data set collection.
  - The minimum set of identification elements required by the PDS standards is sufficient to populate a high-level database.
  - Data preparers will choose additional identification elements from the *Planetary Science Data Dictionary* (PSDD) to support present and future cataloging and search operations.

Label Contents

- Identification Data Elements
  - Required Data Elements for Spacecraft Science Data Products

    DATA_SET_ID
    PRODUCT_ID
    INSTRUMENT_HOST_NAME
    INSTRUMENT_NAME
    TARGET_NAME
    START_TIME
    STOP_TIME
    SPACECRAFT_CLOCK_START_COUNT
    PRODUCT_CREATION_TIME

  - Required Data Elements for Earth-based Science Data Products

    DATA_SET_ID
    PRODUCT_ID
    INSTRUMENT_HOST_NAME
    INSTRUMENT_NAME
    TARGET_NAME
    START_TIME
    STOP_TIME
    PRODUCT_CREATION_TIME

Label Contents

- Descriptive Data Elements
  - PDS recommends inclusion of additional data elements related to specific types of data. These descriptive elements should include any elements needed to interpret or process the data objects or which would be needed to catalog the data product to support potential search criteria at the product level.
  - Recommendations for descriptive data elements to be included come from the PDS mission interface personnel as well as the data producer's own suggestions. These additional data elements are selected from the *Planetary Science Data Dictionary*.

Label Contents

- Data Object Definitions
  - The PDS requires a separate data object definition within the product label for each object in the product, to describe the structure and associated attributes of each constituent object. Each object definition, whether for a primary or a secondary object, must have a corresponding object pointer.
  - The PDS has designed a set of standard data object definitions to be used for labeling products. Among these standard objects are those designed to describe structures commonly used for scientific data storage. The complete set of PDS object definition requirements, along with examples of product labels, is provided in Appendix A of the *PDS Standards Reference*.

Label Contents

- END Statement
  - The END statement ends a PDS label.
  - Where required by an outside agency, the END statement may be followed by one or more SFDU labels.

Validation Tool
Requirements and Design

Sean Hardman

Paul Ramirez

Requirement Overview (1 of 2)

- Scope
  - The scope of the tool specifically focuses on PDS label and product validation, as directed by the PDS Management Council on October 5, 2005 at the Management Council face-to-face meeting.
  - Other aspects of validation (e.g. catalog, volume and data set validation) will be covered in future phases.
- References
  - Planetary Data System (PDS) Validation Tool Requirements, April 1, 2006, Version 1.2.
  - Planetary Data System (PDS) Level 1, 2 and 3 Requirements, May 2006.
  - Planetary Data System (PDS) Standards Reference, March 20, 2006, Version 3.7, JPL D-7669, Part 2.
  - Planetary Science Data Dictionary Document, August 28, 2002, Planetary Data System (PDS), JPL D-7116, Rev E.
  - Tools Survey, April 2005.

Requirements Overview (2 of 2)

- General
  - Requirements pertaining to the identification and specification of PDS data products to be validated by the tool.
- Syntactic Validation
  - Requirements ensuring the grammar of the PDS label is compliant (i.e., the grammar used in labels conforms to the ODL specification, as adopted by the PDS).
- Semantic Validation
  - Requirements ensuring the structure of the PDS label is compliant (i.e., the structure of the objects, groups, keywords, and keyword-values, used in labels, conforms to the Planetary Science Data Dictionary (PSDD) specification).
- Content Validation
  - Requirements ensuring that the PDS label accurately describes the data object (i.e., the objects, groups, keywords, and keyword-values in the description accurately describe the structure used within the data object).
- Reporting
  - Requirements pertaining to reporting the results of validation.

Requirements Status

- On February 23, 2006, the EN held a telecon with the PDS Technical Staff to review the Requirements for the Validation Tool.
  - Reviewed version 1.1 of the Validation Tool Requirements Document.
  - 66 RFAs (55 distinct) were authored as a result of the review.
  - Version 1.2 was prepared and distributed for comment generating another 22 RFAs.
- On June 1, 2006, the EN held a telecon with the PDS Technical Staff to discuss Requirements Status and Preliminary Design of the Validation Tool.
  - A number of issues with regard to the requirements were discussed.
  - The Test Plan and Schedule were also discussed.
  - The preliminary design including a state chart for validation was presented.
  - A number of action items were assigned.
- Current RFA Status
  - 88 Total (60+ Distinct)
    - 52 Open
    - 04 Addressed
    - 02 Tabled
    - 30 Closed

Design Overview

- The preliminary design has focused on aspects of the Validation Tool not likely to change during requirements finalization.
  - Syntactic and semantic validation based on chapter 12 of the PDS Standards Reference.
  - The structure and content of a PDS label.
  - The format and content of a PDS compliant data dictionary.
- Supporting technologies have been incorporated where possible to reduce the overall effort of building the Validation Tool.

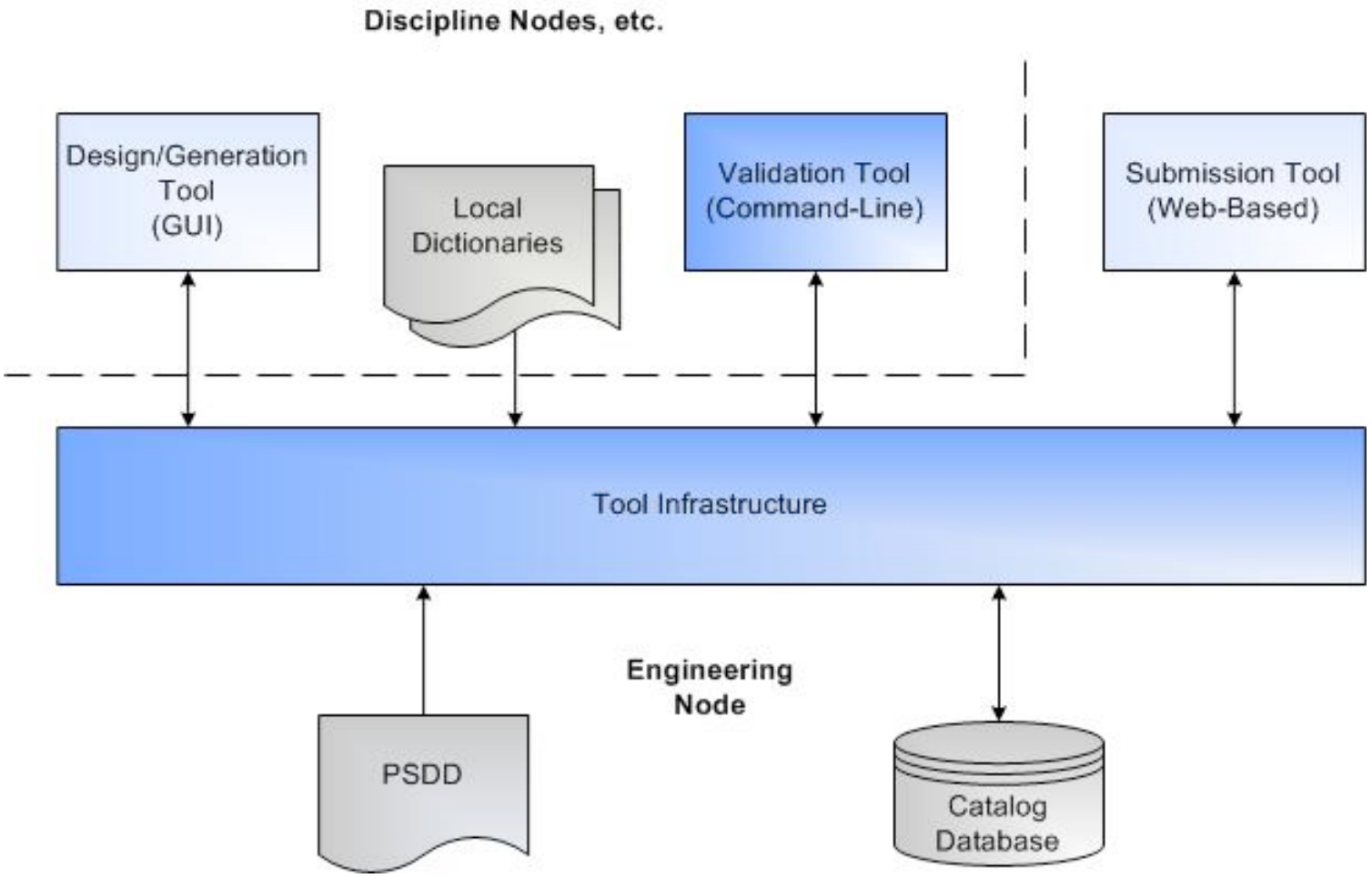
Supporting Technology (1 of 2)

- Java 2 SDK, Standard Edition, Sun Microsystems.
  - Java support is available on the PDS supported platforms (e.g. Linux, Solaris, Windows and Mac OSX).
  - Enables support for additional platforms without additional effort.
  - The tool will be written entirely in Java using version 1.4 for compatibility with older environments.
- Commons Command Line Interface (CLI), Apache Software Foundation.
  - An API for processing command line interfaces.
  - Alleviates the need to develop a command-line parser.
  - Provides for standard and consistent command-line interfaces.
- LOG4J, Apache Software Foundation.
  - An API that allows for control of which log statements are output with arbitrary granularity.
  - Alleviates the need to develop an error message handler.
  - Aides in meeting the Validation Tool's numerous and specific reporting requirements.

Supporting Technology (2 of 2)

- ANother Tool for Language Recognition (ANTLR), Terence Parr.
  - A language tool that provides a framework for constructing recognizers, compilers, and translators from grammatical descriptions.
  - Alleviates the need to develop a label parser from scratch.
  - Provides a structured environment for developing a parser.
- JUnit, Erich Gamma and Kent Beck.
  - A regression testing framework used by the developer who implements unit tests in Java.
  - More on this when Unit Testing is discussed.
- Maven, Apache Software Foundation.
  - A software project management and comprehension tool that can manage a project's build, reporting and documentation aspects.
  - This tool is essentially a glorified “make” allowing developers to better manage the build process.

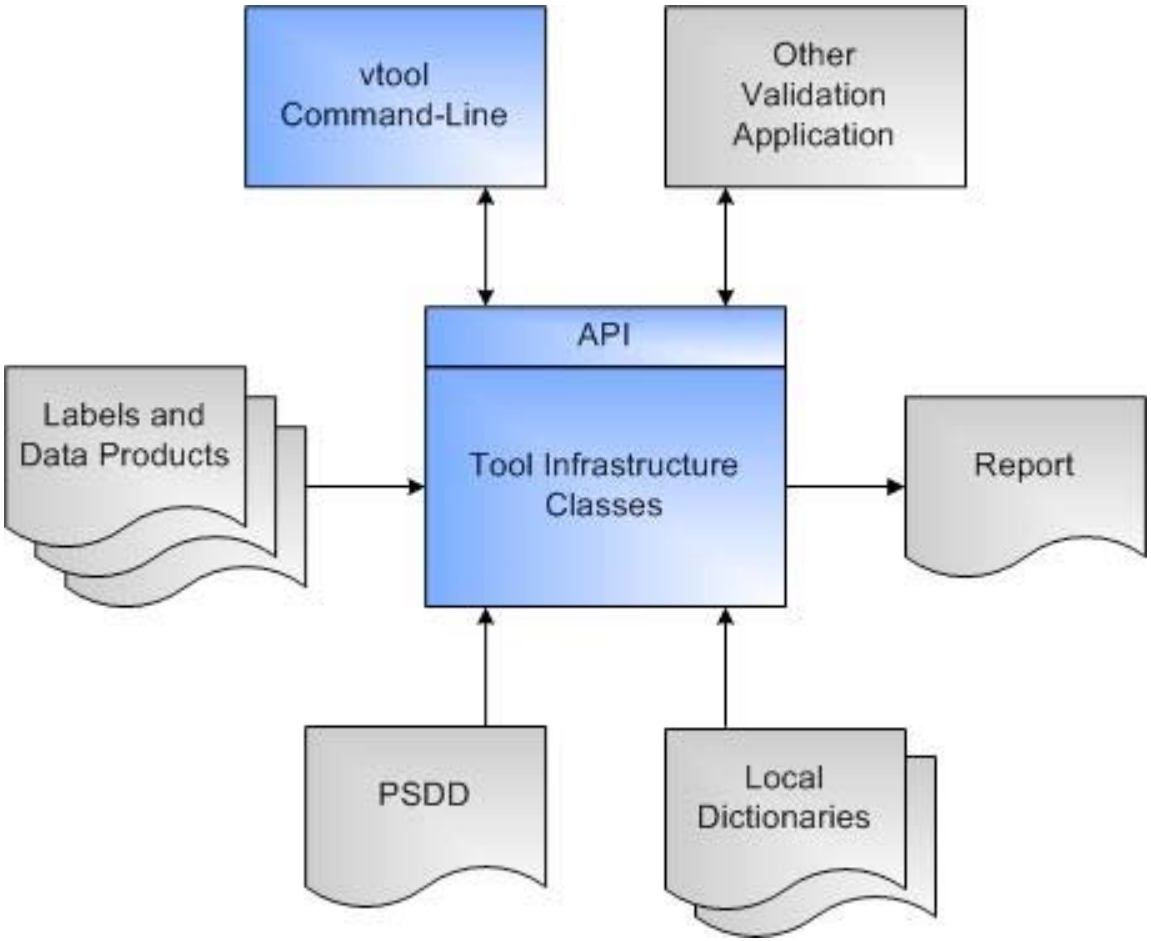
Tool Architecture



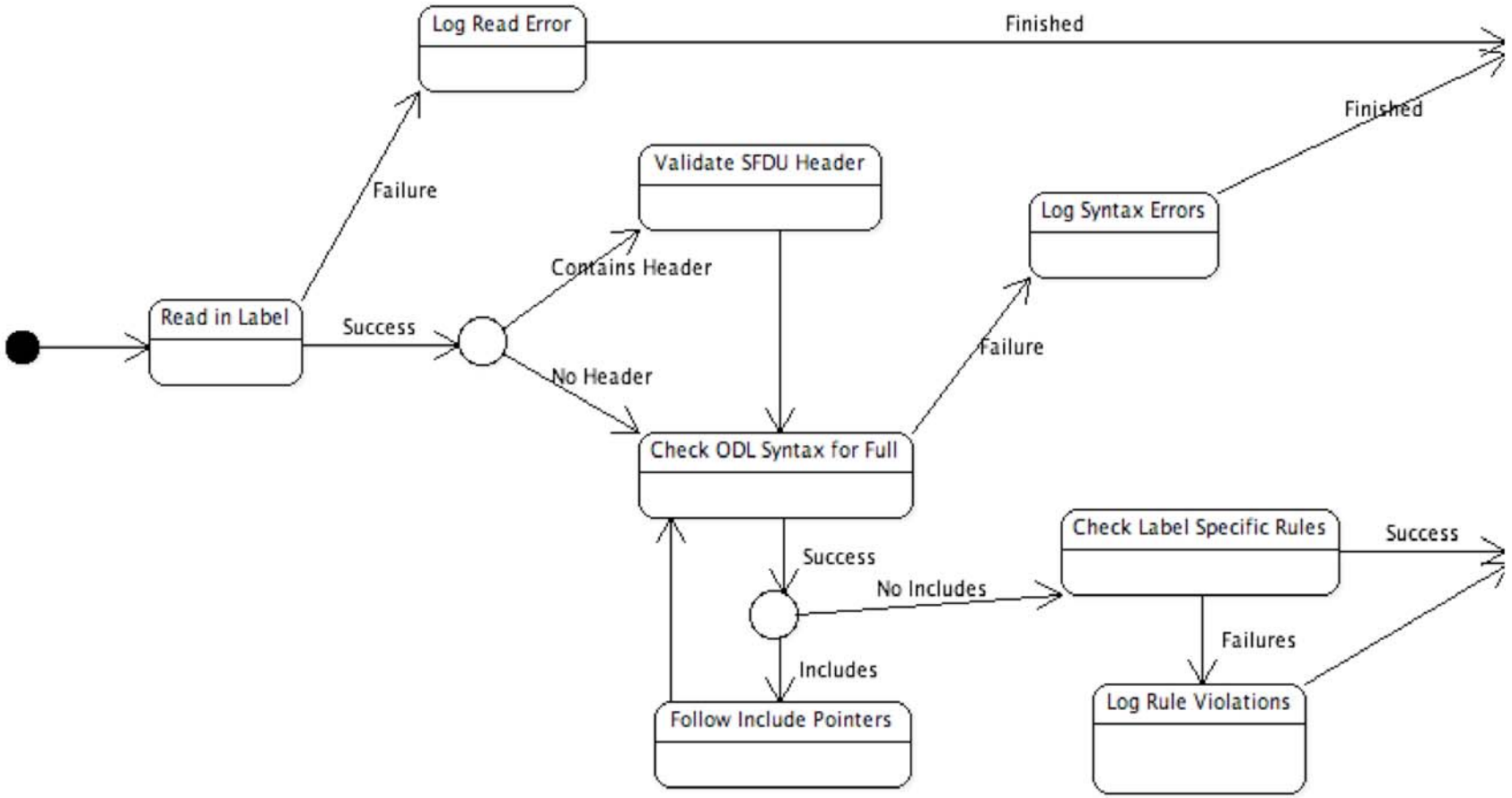
Tool Architecture Description

- Actual and Proposed Tools
  - Design/Generation Tool (Initial Requirements Phase)
    - GUI application intended for use at the Discipline and Data Nodes for designing PDS labels and label templates.
  - Validation Tool (Design and Implementation Phase)
    - Command-line application intended for use at the Discipline and Data Nodes for preparing PDS archival products.
  - Submission Tool (Proposal Phase)
    - Web-based application hosted at the Engineering Node intended for use by the Discipline Nodes for submitting PDS archival products to the PDS Catalog.
- Tool Infrastructure
  - Will provide a single interpretation of the PDS standards for better coordination between the design, validation and submission phases.
  - Will provide common functions for label parsing, dictionary management and reporting.

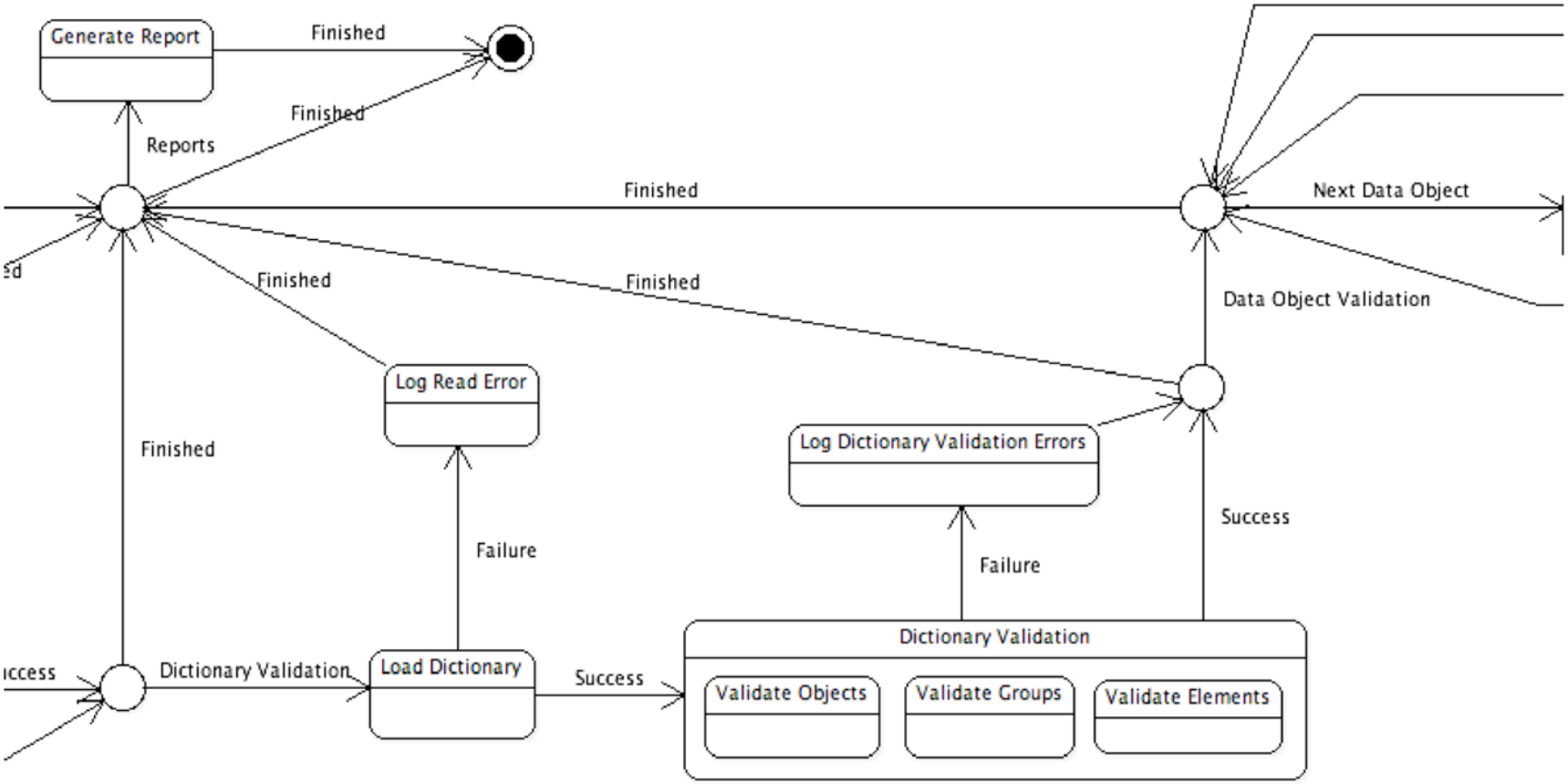
Validation Tool Architecture



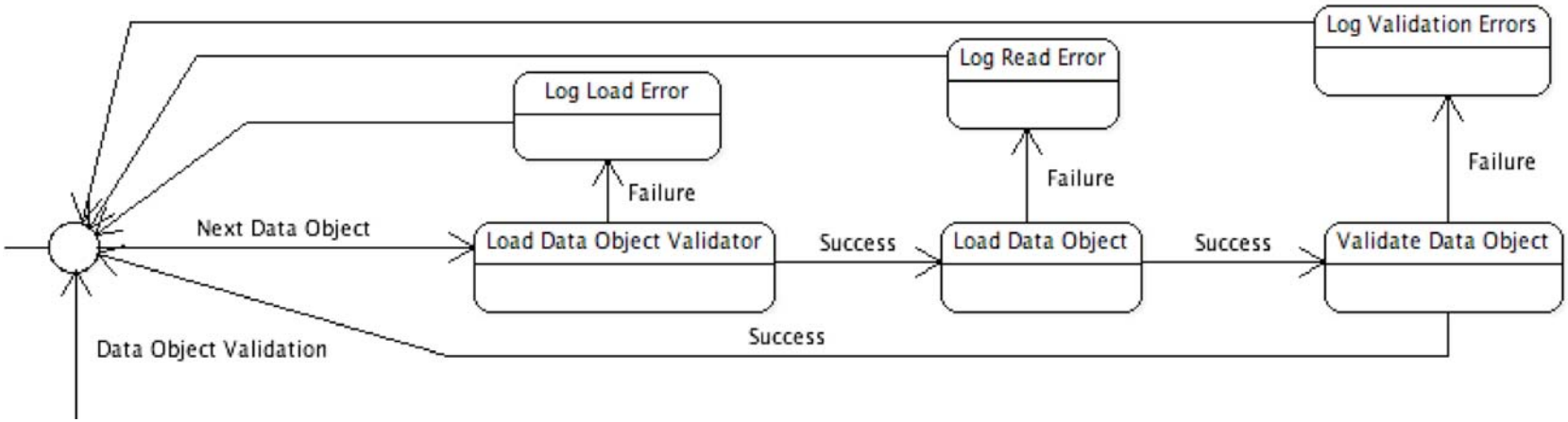
Software Design
State Chart (1 of 3)



Software Design
State Chart (2 of 3)



Software Design
State Chart (3 of 3)



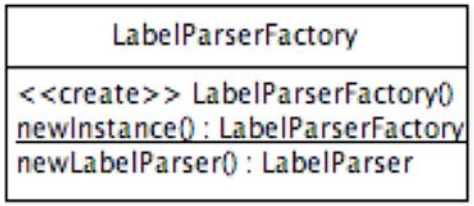
Software Design
State Chart Descriptions (1 of 2)

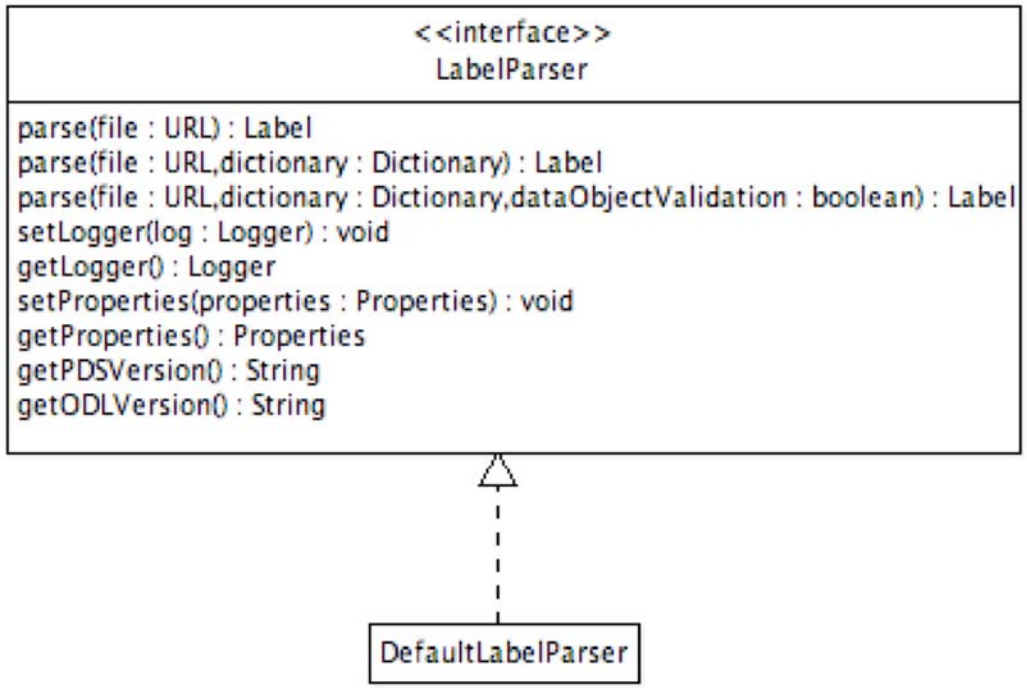
- Read in Label
  - Locate and read in the specified file.
- Validate SFDU Header
  - If a header is present, validate as specified in chapter 16 of the PDS Standards Reference.
- Check ODL Syntax
  - Validate to the grammar specified in chapter 12 of the PDS Standards Reference.
- Follow Include Pointers
  - Locate and include the contents of files referenced by pointers.
- Check Label Specific Rules
  - If this is a data product label, validate it against the associated rules.
- Load Dictionary
  - Locate and read in the specified dictionary.
  - Multiple dictionaries will be supported in following versions.

Software Design
State Chart Descriptions (2 of 2)

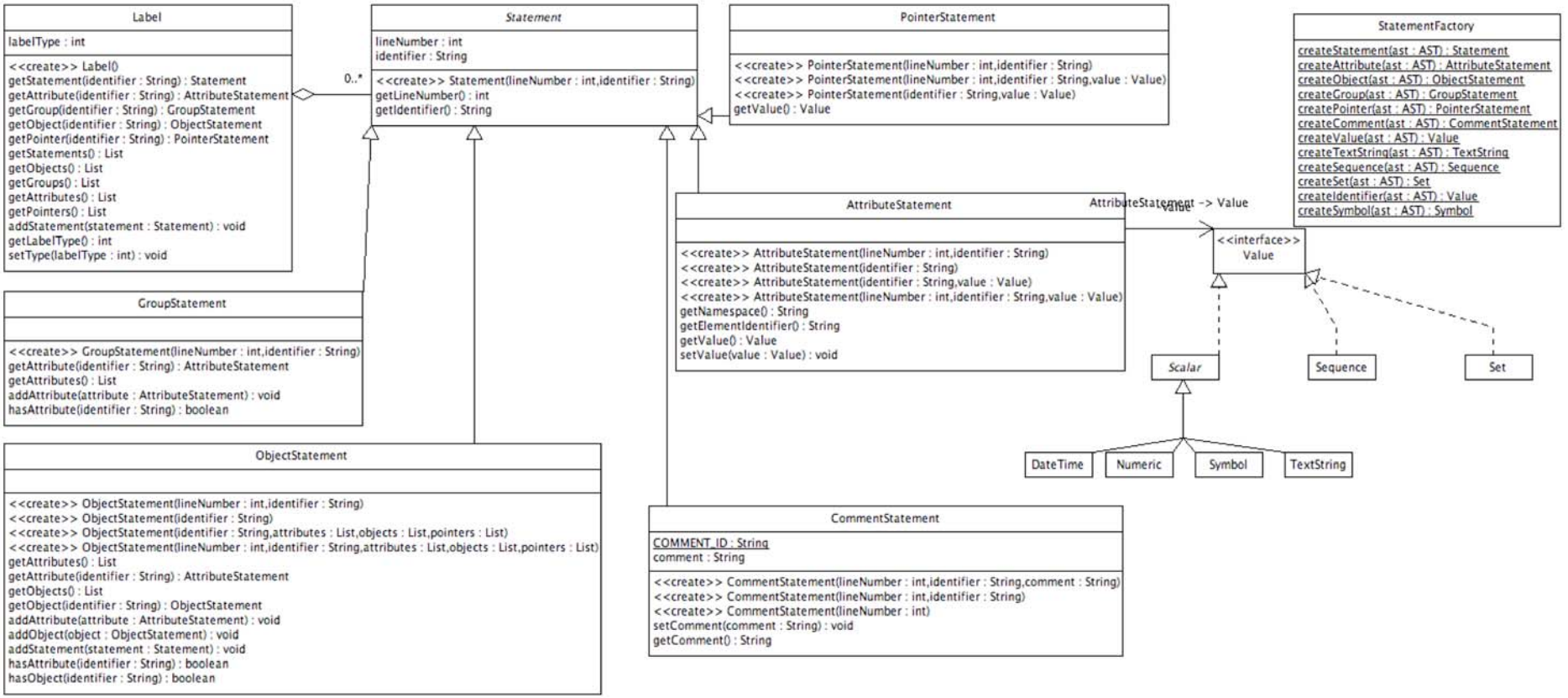
- Dictionary Validation
  - Validate all objects, groups, and elements against loaded dictionary definitions.
- Load Object Validator
  - Depending on the type of object to validate, an appropriate class will be loaded.
- Load Data Object
  - Read in the bytes that make up the data object.
- Validate Data Object
  - Validate the object against its description using the loaded data object validator.
- Generate Report
  - Creates a report according to the specified reporting options.

Software Design
Class Diagram - Label Parser





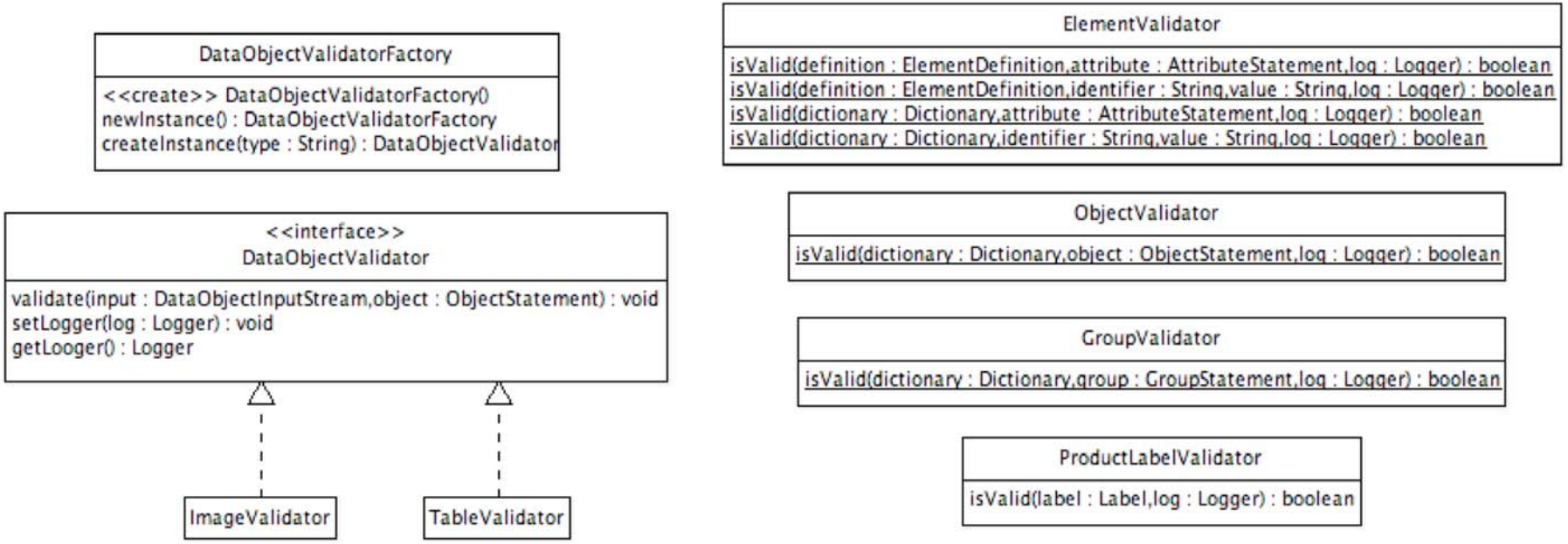
Software Design
Class Diagram - Label Structure



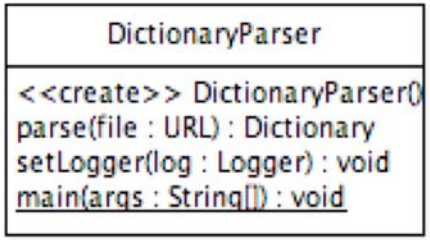
Software Design
Class Description - Label

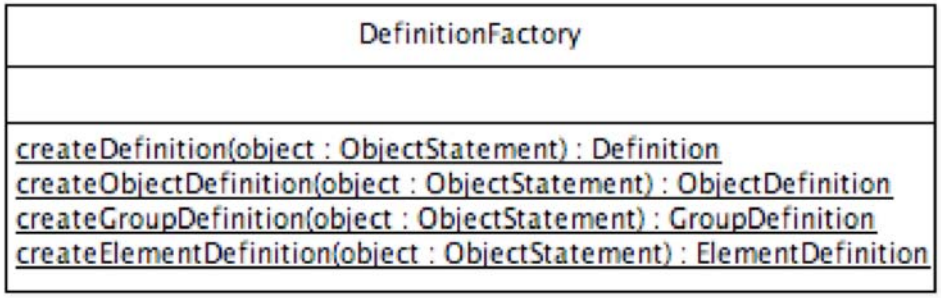
- A PDS Label consists of one or more statements.
- Object Statement
  - Example: OBJECT = IMAGE … END_OBJECT = IMAGE
  - May contain Object statements.
  - May contain Pointer statements.
  - May contain Attribute statements.
- Group Statement
  - Example: GROUP = SHUTTER_TIMES … END_GROUP = SHUTTER_TIMES
  - May contain Attribute statements.
- Pointer Statement
  - Example: ^STRUCTURE = “TABLE.FMT”
- Attribute Statement
  - Example: TARGET_NAME = IO
  - Contains a value:
    - Scalar Value (Text String, Numeric, Date/Time or Symbol)
    - Sequence Value
      - May contain multiple Sequences.
    - Set Value
      - May contain multiple Sets.
- Comment Statement
  - Example: /* Image Description */

Software Design
Class Diagram - Label and Data Object Validators

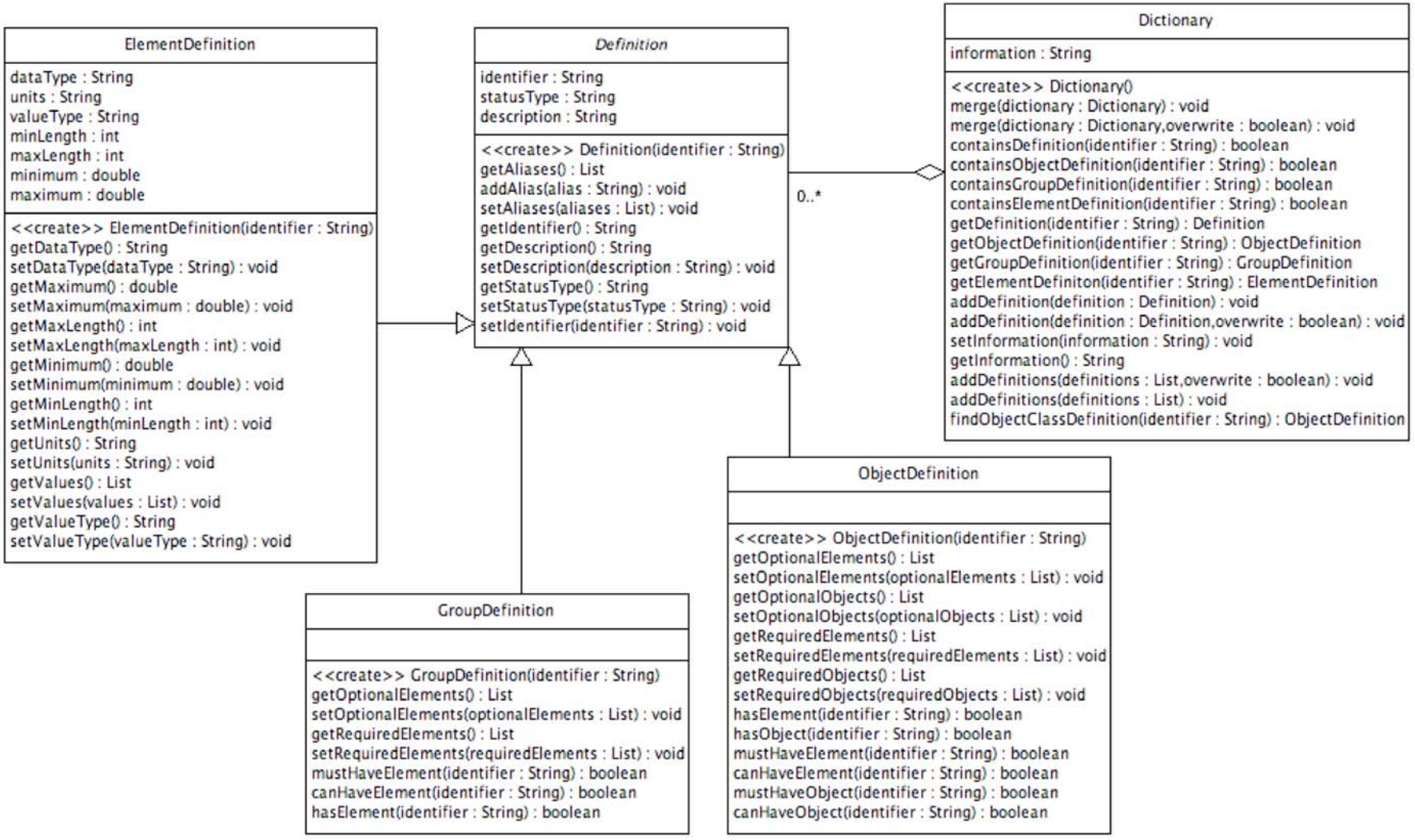


Software Design
Class Diagram - Dictionary Parser

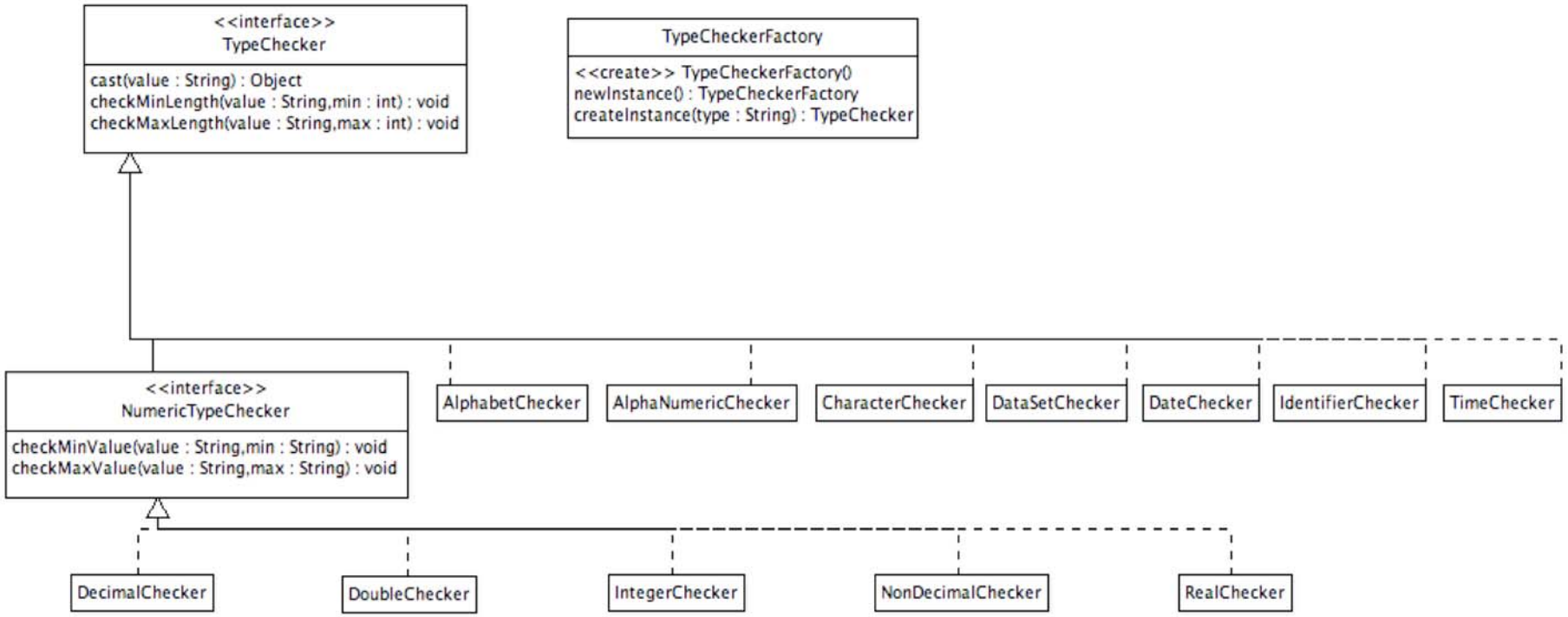




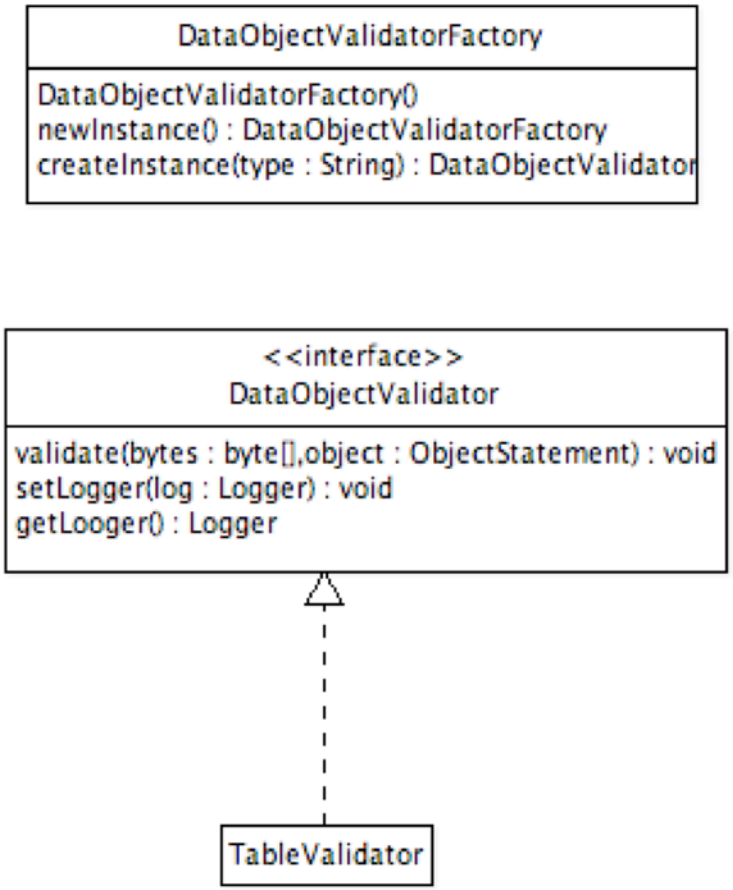
Software Design
Class Diagram - Dictionary



Software Design
Class Diagram - Type Checker



Software Design
Class Diagram - Data Object Validator



Interfaces

- Application
  - Command-Line
    - Proposed, obsolete and questionable options are detailed in the following slides.
  - Application Program Interface (API)
    - Under construction.
- Dictionary
  - As specified in the command-line options, one or more PDS compliant data dictionaries can be referenced.
  - This assumes one PSDD and zero to many local data dictionaries.
  - All specified dictionaries will be merged into one master dictionary for validation.
- Report
  - Still working on the report formats.

Interfaces
Application - Command-Line (1 of 2)

- Proposed Options
  - Validation
    - **Target**: Specifies the file(s) to be validated. Accepts multiple entries, directories and wild cards.
    - **Recursive**: Specifies whether specified directories should be traversed recursively. Default is yes.
    - **Ignore Directories**: Specify a text file containing directories to ignore.
    - **Ignore Files**: Specify a text file containing file extensions to ignore.
    - **Follow Pointers**: Specify whether files referenced by pointers are verified for existence and included for parent label validation. Default is yes. The alternative is to validate as partial labels.
    - **Include Directory**: Specify the path to search for pointer files. Default is the current directory.
    - **Dictionary**: Specifies the PDS compliant data dictionary(s) to be referenced for validation. Assumes the full version and not the index. If not provided, dictionary validation is not performed.
    - **Aliases**: Specify whether aliases are allowed. Default is yes.
    - **Data Objects**: Specify whether data objects are validated. Default is yes.

Interfaces
Application - Command-Line (2 of 2)

- Proposed Options (cont)
  - Reporting
    - **Report**: Specifies the report file specification.
    - **Detail**: Specifies report detail (Verbose, Summary or Minimal). Default is Verbose.
    - **Severity**: Specifies the message severity level and above to include (Debug, Info, Warn, Error or Fatal). Default is Info.
    - **Format**: Specifies the format of the report (Human or Machine Readable). Default is Human.
    - **Max Errors**: Specifies the maximum number of errors to report. Default is 300.
  - Miscellaneous
    - **Help**: Display application usage (command-line arguments).
    - **Version**: Display application version.

Code Walk-through Guidelines

Sean Kelly

Summary (1 of 3)

- Code reviews are the best tool for discovering defects and improving code

- "Ownership" of code is unhealthy
  - Resulting from cognitive dissonance
  - Egoless programming

Summary (2 of 3)

- You are not your code
  - You will make mistakes
- Authority comes from knowledge
  - Review, don't rewrite
- Be kind to the coder
  - Be cruel to the code

Summary (3 of 3)

- Code reviews must be orthogonal to performance reviews
  - Management may reward only for participation
  - Never relate performance to content of review

Validation Tool Code Walk-through

Paul Ramirez

Source Code

- See the source code attachment from the review package.
- Source Tree (pds-tools/trunk/)
  - src/conf/
    - This directory will contain any necessary configuration files.
  - src/java/gov/nasa/jpl/pds/tools/
    - dict/
      - This directory contains Java source code for parsing and representing the dictionary.
    - label/
      - This directory contains Java source code for parsing, representing and validating labels.
    - object/
      - This directory contains Java source code for reading and validating data objects.
  - src/resources/grammar/
    - This directory contains the source code for the label parsing grammar utilized by ANTLER.
  - src/test/
    - This directory will contain source code for the jUnit test cases.
  - src/testdata/
    - This directory contains the initial effort at building the regression test suite.
  - xdocs/
    - This directory will contain the documentation.

Test Approach

Emily Law

Test Plan

- Development Testing
  - Unit Testing: Tests performed at the code-level.
  - Integration Testing: Tests performed at the application-level.
- Beta Testing
  - A phased approach involving Node participation.
- Acceptance Testing
  - Final integration testing to be performed for acceptance.
- Documentation
  - Test Plan
    - The plan for testing the Validation Tool will be incorporated into the Engineering Node Test Plan.
  - Test Procedures
    - Procedures will be developed detailing the steps to be performed, test data to be used and the results expected.
  - Test Reports
    - Reports will be prepared for and made available for each release of the Validation Tool.
  - Anomaly Tracking
    - Bug reports will be tracked and reported on at release.

Test Plan
Development Testing - Unit Testing

- The goal of unit testing is to isolate each part of the program and show that the individual parts are correct.
- A test case is developed to test the interface and functionality of a single class.
- Test cases are exercised at build time allowing for immediate detection of coding anomalies.
- Test cases are included with the source code providing a good source of documentation and enabling on-site testing.
- Test cases for the Validation Tool will be built and managed with Junit (testing framework).

Test Plan
Development Testing - Integration Testing

- For now, integration is limited to integration of the classes into a single command-line program.
- A regression test suite will be built supported with documented test data.
  - Each test case will include test data (e.g. PDS label), a description of the scenario being tested and an example report/result.
  - A procedure will be put in place for accepting test cases from the Nodes.
  - Test cases will be captured and managed in the source tree.
  - Where feasible, this test suite will be automated.
- Cross-platform tests will be performed on PDS-supported platforms.
- Installer package tests will be performed ensuring proper installation.

Test Plan
Beta Testing

- Make integration & test build with test procedures available to Node personnel for local testing.
- Implement a phased and iterative approach, initially including three Nodes (ATMOS, GEO and PPI) with increased participation in subsequent phases.
- Provide a method for accepting test cases from the Nodes to be included in the regression test suite.
- Provide a method for tracking bug reports from the Nodes.

Test Plan
Acceptance Testing

- Test cases generated by the Nodes will be fully incorporated into the final regression test suite.
- The final regression test suite will be performed on acceptance build.

Configuration Management

- Source Code Version Control System
  - Source Code (Java, XML, etc.)
  - Unit Test Cases
  - Regression Test Data
  - Online Documentation
  - Open Source Dependencies
- Document Management System
  - Requirements Artifacts
  - Design Artifacts
  - Test Plan, Procedures and Reports
- PDS Web Site
  - Binary Distributions
  - Source Distributions

Questions
Wrap Up / Action Items